УДК 004.043

О. В. Шпортько

ВИКОРИСТАННЯ ПАЛІТРИ ДЛЯ ГРУПОВОГО СТАТИСТИЧНОГО КОДУВАННЯ RGB-ЗОБРАЖЕНЬ БЕЗ ВТРАТ

The possibility of using palette for the efficiency increase of lossless entropy compression of RGB-images was substantiated. The algorithm of compression realization with the partition of color spectrum of RGB model on nonintersecting parallelepipeds was described. The results of program application, developed using the offered algorithm, for the compression of the ACT image set are presented.

Обґрунтовано можливість використання палітри для підвищення ефективності ентропійного кодування RGB-зображень без втрат. Описано алгоритм реалізації такого стиснення з розбиттям спектра кольорів моделі RGB на паралелепіпеди, що не перетинаються. Наведено результати застосування програми, розробленої з використанням запропонованого алгоритму, для стиснення зображень набору ACT.

У сучасному світі зображення є невід'ємною складовою мультимедійної інформації, що найчастіше зберігається на електромагнітних і цифрових носіях та передається каналами зв'язку. Основними критеріями ефективності форматів графічних файлів для програм, що використовують для перегляду зображень (web-браузерів, провідників, різноманітних довідників та енциклопедій), є показники стиснення та час декодування. Тому проблема підвищення ефективності стиснення зображень не втрачає своєї актуальності останні десятиліття і, ймовірно, не втратить у найближчому майбутньому. Всі алгоритми стиснення зображень поділяють на два основні класи: з втратами та без втрат. Алгоритми з втратами забезпечують значно кращі показники стиснення, тому їх використовують для зберігання більшості фотореалістичних зображень, хоча вони й відтворюють образи з незначними відхиленнями для ока людини. Алгоритми без втрат виконують стиснення значно слабше, проте відтворюють образи, ідентичні оригінальним, що є визначальним чинником для збереження дискретно-тонових зображень. У цій статті пропонуємо алгоритм стиснення зображень без втрат, що можна використовувати як самостійно, так і у поєднанні з відомими алгоритмами.

Аналіз останніх досліджень і публікацій. Будь-яке стиснення даних можливе, насамперед, завдяки зменшенню надлишковостей [5]. У зображеннях надлишковості виявляються найчастіше між сусідніми пікселами, що мають, як правило, близькі кольори, між однаковими фрагментами та між переважаючими інтенсивностями компонент пікселів зображення. Тому більшість сучасних форматів графічних файлів після стиснення даних зображень контекстно-залежними алгоритмами з втратами чи без втрат, що усувають надлишковості перших двох типів, застосовують один з контекстно-незалежних статистичних алгоритмів (найчастіше арифметичний чи Хафмана) для усунення надлишковості третього типу [3]. Ідея використання статистичних алгоритмів полягає у заміні елементів з більшою частотою послідовностями меншої кількості бітів, ніж для елементів з меншою частотою. Середня довжина коду елемента (в бітах) після застосування статистичного алгоритму, згідно з теоремою Шеннона [2], має наближатися до **ентропії джерела**

$$H = -\sum_i p(s_i) \times \log(p(s_i)), \quad (1)$$

де $p(s_i)$ – ймовірність появи елемента s_i (логарифм тут і надалі береться за основою 2). Ентропія джерела зменшується у разі збільшення нерівномірності розподілу ймовірностей між елементами. Нехай кожен з елементів s_i трапляється N_i разів у послідовності довжиною $N = \sum_i N_i$. Тоді $p(s_i) = N_i / N$ і загальна довжина закодованих даних, враховуючи (1), наближається до значення

$$N \times H = N \log(N) - \sum_i N_i \log(N_i). \quad (2)$$

Зменшити ентропію джерела під час обробки зображень найчастіше намагаються за допомогою статичних чи динамічних **предикторів** [7, 2], які прагнуть, використовуючи значення відомих сусідніх компонентів, спрогнозувати (змоделювати) значення чергового компонента зображення. У процесі використання цієї технології обчислюють і надалі кодують **відхилення** чергового компонента від прогнозованого предиктором значення. У загальному випадку процес застосування предиктора для кожного компонента піксела у вузлі (*i*, *j*) записують формулою

$$\Delta_{ij} = F_{ij} - \text{predict}_{ij}, \quad (3)$$

де F_{ij} – значення компонента до застосування предиктора; Δ_{ij} – значення компонента після застосування предиктора; predict_{ij} – значення предиктора для обраного компонента. Оскільки сусідні пікселі зображення мають, як правило, близькі кольори і тому близькі значення відповідних компонентів, то часто значення предиктора збігатиметься зі значенням чергового компонента, найчастіше – буде близьким до цього значення і рідко буде значно відрізнятися від нього. Тобто більшість значень Δ_{ij} виявляються близькими до 0. Такий перерозподіл частот значень значно підвищує нерівномірність розподілу і тому зменшує ентропію джерела (1), а, отже, і довжину закодованої послідовності (2).

Більшість сучасних графічних форматів для зберігання кольору кожного піксела використовують три компоненти [5]. Наприклад, у найпопулярнішій кольоровій моделі RGB – це яскравості червоної (R), зеленої (G) та синьої (B) компоненти. У цьому випадку предиктори, згідно з (3), застосовують до значень кожної компоненти окремо, а статистичне ентропійне кодування, згідно з (1), виконують над результатами дії предикторів усіх компонентів.

Метою дослідження є обґрунтування можливості та опис алгоритму **попіксельного** (а **не покомпонентного**) застосування ентропійного кодування до результатів дії предикторів (3) над компонентами RGB-зображень з використанням палітри та рівномірного кодування [1]. Оскільки рівномірне декодування виконується значно швидше від ентропійного, то така оптимізація дасть змогу прискорити роботу декодера загалом, а раціональне формування палітри дасть змогу, крім цього, покращити коефіцієнт стиснення (КС – це процент зменшення початкового розміру файла) префіксних кодів, що найчастіше використовують на практиці [2, 3, 5].

Результати дослідження. Безпосереднє попіксельне ентропійне кодування результатів дії предикторів, згідно з (3), малоефективне, оскільки значення пікселів відхилень повторюються рідко і можуть бути в межах $[0; 255^3]$. Тому ми пропонуємо **спочатку перетворити значення кольорів пікселів відхилень з використанням палітри** за таким алгоритмом:

1. Згрупуємо кольори пікселів відхилень по паралелепіпедах (тут і надалі – прямокутних), що не перетинаються;

2. Збережемо в палітрі найменші значення компонент пікселів, що належать кожному з утворених паралелепіпедів та їх розміри по кожній осі;
3. Подамо значення кольорів кожного піксела відхилень у вигляді індекса паралелепіпеда, якому він належить, та зміщення по кожній координаті в середині паралелепіпеда.

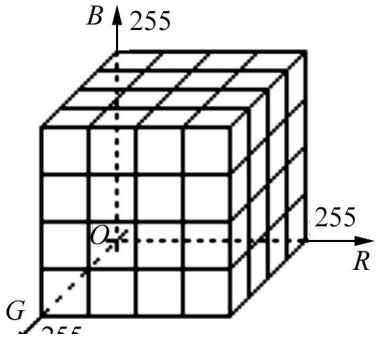
Таке перетворення дає змогу **замість ентропійного кодування** трьох компонент пікселів відхилень **використати групове кодування: ентропійне кодування** лише **індексів відповідних паралелепіпедів в палітрі** та **рівномірне кодування зміщень пікселів відхилень по кожній координаті в середині цих паралелепіпедів**.

Розглянемо тепер питання формування палітри. Обмежимо максимальну кількість кольорів палітри значенням 256, оскільки більші палітри хоча й можливі, але на практиці не використовуються, адже збільшення розміру палітри призводить до збільшення розмірності індексів кольорів кожного піксела в палітрі. Зрозуміло, що формування палітри, тобто, у нашому випадку, розбиття простору кольорів RGB (на рис. 1 – великий куб) на паралелепіпеди, має виконуватися достатньо швидко та забезпечувати високий КС. Тому реалізуємо його в **два** етапи:

1. Спочатку виконаємо **швидке початкове розбиття** простору кольорів RGB **на паралелепіпеди, що не перетинаються**;
2. Потім **поділимо отримані паралелепіпеди** так, **щоб максимізувати КС**.

Початкове розбиття простору кольорів RGB на паралелепіпеди, що не перетинаються, виконаємо згідно з алгоритмом:

1.1. Розіб'ємо множину допустимих значень кольорів на паралелепіпеди максимального фіксованого розміру [4], що не перетинаються між собою (наприклад, для RGB – це множина кубів). Оскільки заздалегідь невідомо, в яких паралелепіпедах містяться кольори пікселів відхилень обраного зображення, то їх множина має повністю покривати множину допустимих значень кольорової моделі а кількість не може перевищувати максимально допустимої кількості кольорів палітри;
1.2. Визначимо кількість кольорів пікселів відхилень та межі їх знаходження у кожному паралелепіпеді. Звузимо межі кожного паралелепіпеда до меж знаходження кольорів пікселів відхилень у ньому;
1.3. Поєднаємо між собою сусідні паралелепіпеди, якщо їх сукупний розмір не перевищує максимального фіксованого;
1.4. Збережемо в палітрі координати та розмірності лише тих паралелепіпедів, що містять кольори пікселів відхилень.



Як свідчать експерименти, початкове розбиття простору кольорів RGB доцільно виконувати на 64 паралелепіпеди розмірами 64×64×64 значення (див. рис. 1), оскільки більші початкові розміри в кілька разів сповільнюють виконання алгоритму, а менші – значно погіршують КС. Кольори пікселів відхилень реальних зображень від прогнозованих предикторами значень містяться, як правило, лише в частині з 64 початкових паралелепіпедів, переважно в тих, що розташовуються біля країв простору RGB.

Визначимо кількість бітів, необхідних для зберігання рівномірного кодування зміщень пікселів відхилень у кожному паралелепіпеді. Нехай діапазон значень ребра чергового паралелепіпеда до поділу по осі R знаходиться в межах [min$R$; max$R$], по осі G – в межах [min$G$; max$G$], а по осі B – в межах [min$B$; max$B$]. Тоді

для рівномірного кодування зміщень піксела відхилень по кожній координаті в середині паралелепіпеда у двійковій системі числення достатньо [log(max$R$ – min$R$ + 1)] + [log(max$G$ – min$G$ + 1)] + [log(max$B$ – min$B$ + 1)] бітів. Наприклад, для рівномірного кодування зміщень у зазначеному паралелепіпеді розмірами 64×64×64 значення достатньо 18 бітів. Запис індекса в палітрі з 64 кольорів рівномірними кодами навіть без ентропійного кодування потребує 6 бітів, тому для зберігання перетвореного значення кольору піксела відхилення достатньо 24 біти, як і у просторі RGB. Тобто навіть у випадку максимальних розмірів усіх паралелепіпедів (для зображень з хаотичними кольорами пікселів) результат групового кодування перевищить розмір вхідного файла лише на опис цих паралелепіпедів. Наприклад, для опису паралелепіпедів, що не перевищують вищезазначеного максимального, достатньо 33 біти: 24 біти для опису базового кольору паралелепіпеда в палітрі і 9 бітів для опису його розмірів – по три для зберігання кількості бітів максимальних зміщень по кожній осі, значення яких належать діапазону [0; 6].

Підвищити КС групового кодування можна завдяки ефективному розбиттю отриманих паралелепіпедів. З одного боку, навіть безпосереднє розбиття довільного паралелепіпеда навпіл зменшує в отриманих паралелепіпедах діапазон значень по осі поділу вдвічі, а кількість бітів для запису зміщень пікселів відхилень в них – на один. Досягнути додаткового зменшення розмірів паралелепіпедів у результаті поділу можна завдяки врахуванню положення зміщень пікселів у ньому. Проілюструємо це на прикладі RG-проекції зміщень кольорів пікселів відхилень окремого умовного паралелепіпеда (див. рис. 2). Розбиття такого паралелепіпеда площиною, перпендикулярною до осі R, посередині ребра (в проекції – штрих-пунктирна лінія l_1) створить два паралелепіпеди: перший – в діапазоні [0; 15] і другий – в діапазоні [16; 31] по цій осі. Тобто сумарного зменшення діапазону значень від такого поділу не відбудеться. Якщо ж розбити цей паралелепіпед аналогічною площиною по значенню R=7 (в проекції – пунктирна лінія l_2), то можемо створити лівий паралелепіпед у діапазоні [0; 5] та правий – у діапазоні [10; 31] по цій осі, що дасть змогу зменшити сумарний діапазон значень на 4 одиниці.

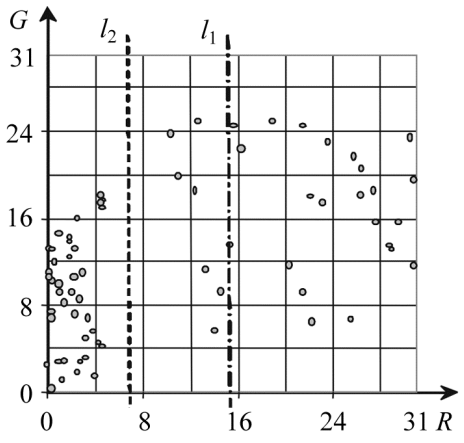


Рис. 2. RG-проекція зміщень кольорів пікселів відхилень окремого умовного паралелепіпеда.

Позначимо через *countPoint* загальну кількість зміщень у паралелепіпеді, через *countR*(j) – кількості зміщень, що мають значення j по осі R, через $\underline{i}$ – праву межу лівого, а через $\bar{i}$ – ліву межу правого з отриманих паралелепіпедів внаслідок поділу по значенню i осі R. Очевидно, що

$$countPoint = \sum_{j=\min R}^{\max R} countR(j)\,,\ \underline{i} = \max_{\substack{j=\min R,\, i \\ countR(j)>0}} j\,,\ \bar{i} = \min_{\substack{j=i+1,\, \max R \\ countR(j)>0}} j\,.$$

Тоді зменшення довжини (кількості бітів) рівномірного коду зміщень пікселів паралелепіпеда внаслідок поділу по значенню i осі R становитиме

$$\Delta'_{R_i} = countPoint \times \lceil \log(\max R - \min R + 1) \rceil -$$
$$- \left(\sum_{j=\min R}^{\underline{i}} countR(j) \times \lceil \log(\underline{i} - \min R + 1) \rceil + \sum_{j=\bar{i}}^{\max R} countR(j) \times \lceil \log(\max R - \bar{i} + 1) \rceil \right). \quad (4)$$

Але розбиття довільного паралелепіпеда призводить до створення нового кольору в палітрі і поділу його пікселів між двома утвореними паралелепіпедами, тобто зменшує нерівномірність розподілу, внаслідок чого збільшує ентропію джерела (згідно з (1)) та загальну довжину коду індексів пікселів відхилень. Тому збільшення дожини ентропійного коду індексів паралелепіпедів (згідно з (2)) внаслідок поділу по значенню *i* осі R становитиме

$$\Delta''_{R_i} = countPoint \times \log(countPo\,\text{int}) -$$
$$- \left(\sum_{j=\min R}^{\underline{i}} countR(j) \times \log\left(\sum_{j=\min R}^{\underline{i}} countR(j) \right) + \sum_{j=\bar{i}}^{\max R} countR(j) \times \log\left(\sum_{j=\bar{i}}^{\max R} countR(j) \right) \right). \quad (5)$$

Зменшення довжини групового коду відхилень унаслідок поділу по значенню *i* осі *R* обчислимо з різниці зменшення рівномірного (4) та збільшення ентропійного (5) кодів:

$$\Delta_{R_i} = \Delta'_{R_i} - \Delta''_{R_i} = countPoint \times \left(\lceil \log(\max R - \min R + 1) \rceil - \log(countPo\,\text{int}) \right) -$$
$$- \sum_{j=\min R}^{\underline{i}} countR(j) \times \left(\lceil \log(\underline{i} - \min R + 1) \rceil - \log\left(\sum_{j=\min R}^{\underline{i}} countR(j) \right) \right) - \quad (6)$$
$$- \sum_{j=\bar{i}}^{\max R} countR(j) \times \left(\lceil \log(\max R - \bar{i} + 1) \rceil - \log\left(\sum_{j=\bar{i}}^{\max R} countR(j) \right) \right).$$

Максимальне зменшення довжини групового коду внаслідок розбиття паралелепіпеда по осі *R* визначимо з умови максимуму цієї величини для всіх можливих поділів по цій осі:

$$\Delta_R = \max_{i=\min R,\, \max R-1} \Delta_{R_i}, \quad (7)$$

а внаслідок довільного поділу паралелепіпеда – з умови максимуму по всіх осях:

$$\Delta = \max\left(\Delta_R, \Delta_G, \Delta_B\right). \quad (8)$$

Максимальне збільшення КС досягається за умови максимального зменшення довжини групового коду. Тому **поділ паралелепіпедів** і, відповідно, розширення палітри **виконуються ітеративно**: під час кожної ітерації визначається максимальне значення Δ для всіх паралелепіпедів згідно з (8) та виконується поділ того паралелепіпеда і тією площиною, де цей максимум досягається. Оскільки на зберігання опису паралелепіпеда витрачається 33 біти, то ітеративний процес припиняється, коли максимальне зменшення довжини групового кодування стане меншим від цього значення або розмір палітри досягне максимально можливого значення.

Розглянемо тепер практичні аспекти реалізації знаходження кращого розбиття паралелепіпеда площинами, перпендикулярними до його осей, на прикладі осі *R*, що дають змогу істотно (до 5%) прискорити виконання обчислень:

1. Для всіх таких *i*, що *countR*(*i*)=0, $\Delta_{R_i} = \Delta_{R_{\underline{i}}}$, тому проводити для них розрахунки зменшення довжини групового коду згідно з (7) не потрібно.

2. Послідовно змінюючи *i*, перераховувати щоразу дві внутрішні суми у (6) не потрібно, адже відносно попередньої перша сума збільшується на *countR*(*i*), а друга – зменшується на цю ж величину.
3. Під час обчислень двійкових логарифмів згідно з (6), які, як правило, не реалізовані на рівні мови програмування, замість класичної формули $\log_2(x) = \ln(x)/\ln(2)$ доцільно використати $\log_2(x) = \ln(x) \times 1{,}4426950409$, що зменшить кількість викликів вбудованих функцій та складність виконання обчислень.
4. Для реалізації функції $\lceil \log(\ ) \rceil$ замість використання вбудованих функцій логарифмування та заокруглення до більшого числа доцільно здійснити лише реагування на порогові значення. Мовою С таке реагування може бути реалізоване, наприклад, так:

```
int countBit(UBYTE4 diapazon)
 {if (diapazon<=1) return 0;
 if (diapazon==2) return 1;
 if (diapazon<=4) return 2;
 if (diapazon<=8) return 3;
 ... }.
```

5. Краще розбиття кожного зі створених паралелепіпедів не залежить від інших розбиттів паралелепіпедів, тому після кожної ітерації алгоритму палітрування додатково розраховувати максимальне зменшення довжини групового коду необхідно лише для двох паралелепіпедів, що утворилися в результаті поділу.

Результати експериментів. Розглянемо результати застосування описаного алгоритму групового статистичного кодування для стиснення файлів стандартного набору ACT 24-бітних зображень, характеристики яких наведено в [7]. Завантажити TIFF-версій файлів зображень цього набору можна, наприклад, з http: //compression.ca/act/act-files.html. Цей набір містить як дискретно-тонові (№ 2, 7) так і неперервно-тонові (всі інші) зображення. Тестування проводили за допомогою модифікованої програми з CD [3] для запису зображень у форматі PNG, яка послідовно опрацьовує дані в такій послідовності: попіксельне віднімання значення зеленої компоненти від червоної та синьої; стиснення зображення контекстно-залежним алгоритмом LZPR [7], який базується на словниковому алгоритмі LZ77 [8] та кодує повторення; застосування предиктора Піфа [3] до кожної компоненти незакодованих пікселів; перетворення значення кольорів незакодованих пікселів відхилень з використанням палітри згідно з описаним алгоритмом; кодування для пікселів відхилень індексів відповідних паралелепіпедів префіксними кодами Хафмана та зміщень рівномірними кодами; проведення аналізу ефективності стиснутих блоків [6].

Таблиця 1. Розміри файлів кодів після застосування різних варіантів палітрування зображень набору ACT, Кб

№ файла	Без поділу паралелепіпедів				З поділом паралелепіпедів			
	Ентропійні коди	Рівномірні коди	Всього	КС, %	Ентропійні коди	Рівномірні коди	Всього	КС, %
1	156	1575	1731	17,61	223	200	423	79,87
2	238	2715	2953	18,47	303	234	537	85,17
3	92	577	669	13,00	229	228	457	40,57
4	136	804	940	18,47	332	154	486	57,85
5	94	561	655	14,82	225	132	357	53,58
6	144	861	1005	12,84	349	217	566	50,91
7	95	1098	1193	18,51	118	89	207	85,86
8	141	864	1005	12,84	343	212	555	51,86
Разом	**1096**	**9055**	**10151**	**16,69**	**2122**	**1466**	**3588**	**70,55**

Таблиця 2. Час кодування/декодування програми для різних варіантів палітрування зображень набору ACT (на комп'ютері з частотою процесора 300 МГц), с

№ файла	Час кодування				Час декодування			
	Без поділу паралелепіпедів		З поділом паралелепіпедів		Без поділу паралелепіпедів		З поділом паралелепіпедів	
	Всього	В т. ч. палітр.	Всього	В т. ч. палітр.	Всього	В т. ч. депалітр.	Всього	В т. ч. депалітр.
1	8,19	2,47	9,50	4,45	5,27	1,27	4,17	0,16
2	13,18	4,17	15,49	6,92	9,07	2,19	6,71	0,17
3	3,02	0,88	5,11	3,08	2,14	0,44	2,19	0,16
4	4,34	1,48	7,25	4,29	3,08	0,60	2,97	0,11
5	2,97	0,94	4,89	2,91	2,09	0,44	2,03	0,11
6	4,45	1,54	7,64	4,61	3,14	0,66	3,07	0,22
7	5,22	1,76	6,04	2,92	3,74	0,87	2,75	0,06
8	4,33	1,53	7,47	4,34	3,13	0,66	3,07	0,17
Разом	**45,70**	**14,77**	**63,39**	**33,52**	**31,66**	**7,13**	**26,96**	**1,16**

Результати безпосереднього застосування алгоритму групового статистичного кодування (без другого та шостого етапів програми) з різними варіантами палітрування наведено в табл. 1, 2. З даних цих таблиць видно, що поділ окремих паралелепіпедів хоча й у середньому сповільнює кодування на 39%, проте прискорює декодування на 15% і, головне, підвищує КС на 53,86%, причому покращення стиснення відбувається внаслідок кардинального зменшення довжини рівномірних кодів.

Як зазначалося вище, описаний алгоритм групового статистичного кодування належить до класу контекстно-незалежних. Тому розглянемо результати його застосування (див. стовпці 3, 7 в табл. 3, 4) після контекстно-залежного алгоритму з аналізом блоків стиснутих даних (всі етапи програми). Для порівняння в цих же таблицях наведені результати тестування програми без палітрування (стовпці 2, 6), популярного архіватора WinRAR 3.00 (стовпці 4, 8), який використовує той самий алгоритм LZ77, та програми ERI 5.1 (стовпці 5, 9), що базується на алгоритмі паралельного сортування блоків [2, 4] та забезпечує максимальний на сьогодні КС по набору зображень ACT (за даними http://www.compression.ru/arctest/act/act-tif.htm).

Таблиця 3. Коефіцієнти стиснення зображень набору ACT різними програмами, %

№ файла	LZPR	LZPR з палітруванням	WinRAR	ERI
1	78,96	82,25	76,06	83,48
2	94,34	94,40	94,37	94,81
3	39,01	40,83	38,49	39,92
4	56,20	58,37	55,16	61,67
5	52,15	53,58	52,02	56,31
6	47,70	51,17	49,09	53,95
7	93,99	93,99	93,92	94,67
8	50,13	52,04	49,00	55,85
Середній	**64,06**	**65,83**	**63,51**	**67,58**
Сукупний	**73,28**	**74,79**	**72,67**	**76,26**

Дані цих таблиць свідчать, що використання палітри після контекстно-незалежного алгоритму хоча й у середньому сповільнило кодування на 15%, але прискорило декодування на 6% та підвищило КС на 1,77%. Програма з груповим статистичним кодуванням хоча й працює значно повільніше від WinRAR, проте

забезпечує у середньому кращі КС. Порівняно з ERI 5.1 тестова програма демонструє у середньому гірші КС, але забезпечує швидше декодування.

Таблиця 4. Час кодування/декодування зображень набору ACT різними програмами (на комп'ютері з частотою процесора 300 МГц), с

№ файла	Час кодування				Час декодування			
	LZPR	LZPR з палітр.	WinRAR	ERI	LZPR	LZPR з палітр.	WinRAR	ERI
1	11,65	14,17	5,88	3,95	3,95	3,79	0,55	4,56
2	23,62	24,28	7,31	5,27	5,17	5,11	2,91	5,05
3	8,07	10,65	1,98	3,51	2,58	2,20	0,44	3,79
4	20,60	23,73	3,08	3,68	3,13	2,91	0,71	4,07
5	10,88	13,73	1,98	2,96	2,25	2,03	0,43	3,08
6	19,93	23,35	3,90	4,17	3,57	3,18	0,66	4,51
7	9,89	10,33	3,18	2,31	2,09	2,15	1,21	1,93
8	18,45	21,59	2,91	4,07	3,35	3,02	0,66	4,34
Разом	**123,09**	**141,83**	**30,22**	**29,92**	**26,09**	**24,39**	**7,57**	**31,33**

Для досягнення кращих КС та прискорення палітрування **в подальшому** планується реалізувати групове статистичне стиснення з використанням арифметичного кодування, розробити контекстно-незалежний спосіб палітрування на основі аналізу гістограми частот компонентів пікселів та контекстно-залежний спосіб палітрування на основі аналізу послідовностей пікселів.

З наведених результатів дослідження можна зробити такі **висновки**:

1. Формування ефективної палітри для групового статистичного кодування можливе лише за умови врахування довжини як ентропійних, так і рівномірних кодів.
2. Групове статистичне кодування дає змогу досягнути кращих КС порівняно з результатами кодування Хафмана внаслідок зменшення надлишковостей кодів.
3. Неперервно-тонові зображення доцільно стискати запропонованим алгоритмом безпосередньо (без попереднього стиснення контекстно-залежним алгоритмом). Така оптимізація хоча й погіршує КС в межах 1%, зате дає змогу прискорити кодування більше, ніж удвічі.
4. Застосування алгоритму групового статистичного кодування з допомогою палітри замість ентропійного кодування дає змогу суттєво зменшити час декодування внаслідок введення рівномірних кодів, і тому може бути рекомендоване для комплексного використання у форматах стиснення зображень без втрат.

1. *Кадач А. В.* Эффективные алгоритмы неискажающего сжатия текстовой информации: Дис… канд. физ.-мат. наук – Ин-т систем информатики им. А.П. Ершова. – Новосибирск, 1997. – 200 с.
2. *Методы* сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М.: ДИАЛОГ-МИФИ, 2003. – 384 с.
3. *Миано Дж.* Форматы и алгоритмы сжатия изображений в действии: Учеб. пособ. – М.: Триумф, 2003. – 336 с.
4. *Ратушняк О. А.* Алгоритмы сжатия изображений без потерь с помощью сортировки параллельных блоков // Тезисы докладов конференции молодых ученых по математике, мат. моделированию и информатике. – Новосибирск, 2001. – С. 48–49.
5. *Сэломон Д.* Сжатие данных, изображений и звука. – М.: Техносфера, 2006. – 336 с.
6. *Шпортько О. В.* Використання альтернативних блоків стиснутих даних у форматі PNG // Комп'ютерні науки та інформаційні технології: Матеріали третьої Міжнародної конференції CSIT'2008. – Львів: ПП "Вежа і Ко", 2008. – С. 149–153.
7. *Шпортько О. В.* Оптимізація використання статичних предикторів у процесі стиснення зображень без втрат // Відбір і обробка інформації. – 2008. – № 28(104). – С. 82–89.
8. *Ziv J., Lempel A.* A universal algorithm for sequential data compression // IEEE Transactions on Information Theory. – May 1977. – Vol. 23(3). – P. 337–343.

Рівненський державний гуманітарний університет

Одержано 20.09.2008